THE LINGUISTIC EXPLORATIONS OF CHILDREN
PLAYING WITH LANGUAGE THROUGH COMPUTER PROGRAMMING

by **SAVALAI VAIKAKUL**
S.B. Biology Massachusetts Institute of Technology 1997

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the **Massachusetts Institute of Technology** June 1999

©1999 Massachusetts Institute of Technology
All rights reserved.

ROTCH

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUN 14 1999

LIBRARIES

Author **Savalai Vaikakul**
Program in Media Arts and Sciences
7 May 1999

Certified by **Seymour Papert**
Professor of Education and Media Technology
Program in Media Arts and Sciences
Thesis Advisor

Accepted by **Stephen Benton**
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

THE LINGUISTIC EXPLORATIONS OF CHILDREN: PLAYING WITH LANGUAGE THROUGH COMPUTER PROGRAMMING

by **SAVALAI VAIKAKUL**

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning on 7 May 1999
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the **Massachusetts Institute of Technology**

ABSTRACT

Children's intuitions about the grammar of their language are resources which children can use to leverage understanding of formal grammatical concepts. In this thesis, I demonstrate how the Logo programming environment can be used to encourage and support children's intuitive explorations in the domain of formal linguistics. Computer programming was used to create a meaningful context in which formal grammatical concepts were introduced to children through the engagement and mobilization of their linguistic intuitions. To initially engage and mobilize children's linguistic intuitions, I made a computer program in which children could play at using their intuitions about the English language to figure out the basis of a turtle character's linguistic judgments. In the context of working to understand how my program was made, children arrived at a meaningful understanding of the formal linguistic concepts I had used to construct my computer program. Furthermore, children personally appropriated the formal linguistic concepts for the purpose of modifying my original program and making their own computer programs about language.

Thesis Advisor **Seymour Papert**
Professor of Education and Media Technology
Program in Media Arts and Sciences

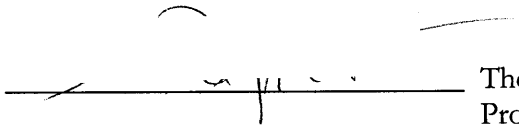
Work supported in part by
the National Science Foundation, the LEGO Corporation, and the Suksapattana Foundation.

THE LINGUISTIC EXPLORATIONS OF CHILDREN: PLAYING WITH LANGUAGE THROUGH COMPUTER PROGRAMMING

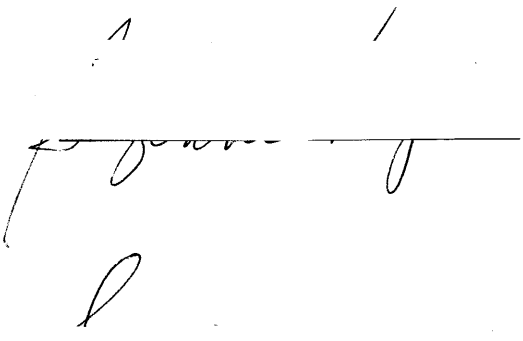
by **SAVALAI VAIKAKUL**

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning on 7 May 1999
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the **Massachusetts Institute of Technology**

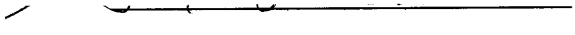
THESIS COMMITTEE



Thesis Advisor **Seymour Papert**
Professor of Education and Media Technology
Program in Media Arts and Sciences



Thesis Reader **Suzanne Flynn**
Professor of Linguistics and Second Language Acquisition
Section Head, Foreign Languages and Literatures
MIT Department of Humanities
Professor of Second Language Acquisition
MIT Department of Linguistics and Philosophy



Thesis Reader **Edith Ackermann**
Senior Research Scientist
Mitsubishi Electric Research Laboratories, Inc.

ACKNOWLEDGMENTS

A very big thank you to my advisor, Seymour Papert, and to my thesis readers, Edith Ackermann and Suzanne Flynn, for their support and encouragement throughout the thesis research and writing process. Another big thank you to Todd Atkins and Arun Tharuvai, for their help with Logo programming and working with children. I would also like to thank the group of children participants and their parents. Additionally, I would like to thank members of the Epistemology and Learning Group. And special thanks to Claudia Urrea, for being the best office mate ever.

A thank you goes out to my friends, Greg Qushair, Mike Dimyan, Carter Malkasian, and Aya Nakamura.

I am very fortunate to be a part of two loving families, who have given me more happiness than I could ever give in return. I would like to thank my mother, Sauvakon Vaikakul, my father, Vatcharin Vaikakul, my sister, Vatcharavee Vicky Vaikakul, and my grandmothers, Malai Vaikakul and Somsuk Lekutai. I would also like to thank my second family – Janet and Tom Ridley, Bryan and Brandon Ridley, and Marguerite Ridley.

I am forever thankful for my husband, Brent Ridley, who has been my best friend for these past six years.

All glory be to God and Jesus Christ Our Savior. I can do all things through Him who strengthens me.

CONTENTS

MOTIVATIONS

In *Mindstorms*, Seymour Papert told a story about Jenny, a 13-year-old 7th-grader, who was working on programming the computer to generate sentences by choosing words on the basis of whether they were nouns or verbs, when she surprised herself with this discovery: "Now I know why we have nouns and verbs," Jenny said.[1]

> For many years in school Jenny had been drilled in grammatical categories [but] she had never understood the differences between nouns and verbs and adverbs. But now it was apparent that [Jenny's] difficulty with [the formal concept of grammatical categories] was not due to an inability to work with logical categories. It was something else. She had simply seen no purpose in the enterprise. She had not been able to make any sense of what [grammatical categories were] about in the sense of what [they] might be *for*.
>
> [A]s she tried to get the computer to generate [sentences], something remarkable happened. She found herself classifying words into categories, not because she had been told she had to but because she needed to. In order to "teach" her computer to make strings of words that would look like English, she had to "teach" it to choose words of an appropriate class ... [From this task] Jenny did more than learn definitions for particular grammatical classes. She understood the general idea that words (like things) can be placed in different groups or sets, and that doing so could work for her.[2]

As a speaker of English, Jenny has intuitions about which English words can be classified as nouns and which English words can be classified as verbs.[3] Through the process of making her computer program, these intuitions and the formal concept of grammatical categories finally became connected for Jenny. Jenny was excited because the formal concept of grammatical categories – which had frustratingly eluded her for so long despite all the drill and practice in school – became meaningful and purposeful. She could use grammatical terms like "nouns" and "verbs" as descriptive tools to articulate her intuitive knowledge about nouns and verbs, for the purpose of constructing a sentence generating system.

---

[1] Papert80, pg. 48.
[2] Ibid., pg. 49.
[3] Chomsky72.

Like Jenny, I too had difficulty understanding grammatical concepts as they were presented in school. English class – like the majority of my classes – offered me nothing that I saw as meaningful or useful. To me, grammatical facts and rules presented in English class were completely divorced from what I knew, and my only motivation for learning was the threat of getting a poor grade. It was in my final year of undergraduate education when I discovered what Jenny had discovered – there *is* a meaning and a purpose to those formal grammatical concepts taught in English class!

My discovery happened in an introductory linguistics course. During the course, we looked at various linguistic phenomena in our everyday language use (such as plurals, pronouns, etc.) and worked to articulate the processes underlying our intuitive production and interpretation of these linguistic forms. Through this experience, I developed connections between the linguistic intuitions inside my head and the formal concepts from my English classes – just as Jenny developed the connection between her intuitive knowledge about nouns and verbs and the formal categories of "nouns" and "verbs" through looking at the natural process of sentence formation and articulating it to the computer in the form of a program.

Jenny and I were able to use our linguistic intuitions to make a connection with formal grammatical concepts, and therefore forge a new relationship with linguistic knowledge, because we were in situations in which we were motivated to think "scientifically." I am using the term "science" to convey a sense of formal activity in which one focuses on the world and its phenomena and works to uncover general truths that comprise a system of knowledge. In her situation, Jenny had to focus on the linguistic phenomenon of sentence formation. In working to articulate a generative system of sentence formation to the computer, Jenny had to harness both the logical power of her intuitions about how words are categorized and the descriptive power of formal grammatical terms. What I had to do in the introductory linguistics course was also to look at a specific linguistic phenomenon and work to articulate a systematic account of the phenomenon by appealing to my linguistic intuitions and the descriptive power of formal linguistic concepts.

Children possess a vast amount of intuitions about the grammar of the language that they speak.[4] The task at hand, then, is *not* the instruction of children about "disembodied" grammatical facts and rules. A much better way to support children's understanding of formal grammatical concepts is to provide opportunities for children to discover how much they already know intuitively, use their existing knowledge to leverage enthusiasm for more knowledge, and encourage further knowledge construction by providing them with rich conceptual tools and activities.

---

[4] Chomsky72.

My general impression is that many educators mistakenly diagnose the difference between children's intuitive ways of thinking and the ways of thinking which are given a privileged status in school as a *problem* which education must *address*. It often appears to be the case that the mission of education is to help children *abandon* their intuitive understandings in order to adopt "the precise, articulated, and systematic understandings characteristic of science and mathematics." [5] This devaluation of children's intuitive ways of understanding contributes significantly to the fact that many children have such a difficult time "connecting" to domains of formal knowledge – like grammar or mathematics – as they are presented in school.

> [Some children] ... compartmentalize "school knowledge", treating it as though it had nothing to do with the knowledge by which [they understand and operate] in "real life." [Some children] ... treat the symbols of formal understanding gained in school as a "closed-system vocabulary", an empty formal language which has no referent in [their] experience at all. [Some children] remain faithful to the understandings [they have] gained in the world outside of the school, behaving in the school as though [they] were unwilling or unable to learn.[6]

Clearly, abandoning intuitive ways of thinking and doing does not facilitate formal ways of thinking and understanding. On the contrary, children's intuitions are resources which children can use to leverage understanding of formal concepts in academic domains. Taking guidance from the story of Jenny and from my own experience in the linguistics class, I believe that an environment in which formal grammatical concepts are introduced to children through the engagement and mobilization of their intuitions about the grammar of their language would prove to be an effective context for formal grammatical concepts to become meaningful and purposeful to children.

---

[5] Schön81, pg. 24.
[6] Ibid., pg. 25.

FOUNDATIONS

Turtle geometry and mathematical knowledge ★

In a chapter entitled "Turtle Geometry: A Mathematics Made for Learning" in *Mindstorms*, Papert wrote about using a computational object called a "Turtle" to enable children to intuitively explore mathematical concepts in a Logo programming environment.[7] He presented his ideas in the context of Euclidean geometry:

> Euclid built his geometry from a set of fundamental concepts, one of which is the point. A point can be defined as an entity that has a position but no other properties – it has no color, no size, no shape. People who have not yet been initiated into formal mathematics, who have not yet been "mathematized", often find this notion difficult to grasp, even bizarre. It is hard for them to relate it to anything else they know. Turtle geometry, too, has a fundamental entity similar to Euclid's point. But this entity, which I call a "Turtle", can be related to things people know because unlike Euclid's point, it is not stripped so totally of all properties, and instead of being static it is dynamic. Besides position the Turtle has one other important property: It has "heading." A Euclidean point is at some place – it has a position, and that is all you can say about it. A Turtle is at some place – it, too, has a position – but it also faces some direction – its heading. In this, the Turtle is like a person – I am here and I am facing north – or an animal or a boat. And from these similarities comes the Turtle's special ability to serve as a first representative of formal mathematics for a child. Children can identify with the Turtle and are thus able to bring their knowledge about their bodies and how they move into the work of learning formal geometry." [8]

[7] Papert80. Logo is a programming language invented by Seymour Papert.
[8] Ibid., pg. 55-56.

Papert noted that children gain two kinds of knowledge when they play with mathematical concepts via the Turtle: mathematical knowledge and "mathetic" knowledge.[9] The mathematical knowledge is Turtle geometry, "a kind of geometry that is easily learnable and an effective carrier of very general mathematical ideas." [10] Mathetic knowledge is "knowledge about learning." [11]

> We introduced Turtle geometry by relating it to a fundamental mathetic principle: Make sense of what you want to learn. Recall the case of Jenny, who possessed the conceptual prerequisites for defining nouns or verbs but who could not learn grammar because she could not *identify* with this enterprise. In this very fundamental way grammar did not make sense to her. Turtle geometry was specifically designed to be something children *could* make sense of, to be something that would resonate with their sense of what is important.[12]

"Turtle geometry is learnable because it is syntonic" – meaning it is "firmly related to children's sense and knowledge about their own bodies." [13] Turtle geometry is also "an aid to learning other things because it encourages the conscious, deliberate use of problem-solving and mathetic strategies." [14] Papert noted that he did not "invent" syntonic mathematics for children – it is a body of intuitive knowledge which children already have.

> [I] have merely given children a way to reappropriate what was always theirs. Most people feel that they have no "personal" involvement with mathematics, yet as children they constructed it for themselves. Jean Piaget's work on genetic epistemology teaches us that from the first days of life a child is engaged in an enterprise of extracting mathematical knowledge from the intersection of body with environment. The point is that, whether we intend it or not, the teaching of mathematics, as it is traditionally done in our schools, is a process by which we ask the child to forget the natural experience of mathematics in order to learn a new set of rules.[15]

---

[9] Papert80, pg. 63.
[10] Ibid., pg. 63.
[11] Ibid., pg. 63.
[12] Ibid., pg. 63.
[13] Ibid., pg. 63.
[14] Ibid., pg. 63.
[15] Ibid., pg. 207.

Language as a problem space for children ★

Naturally, language is a domain of scientific inquiry for children. In fact, children's cognitive activity in the linguistic problem space shares many characteristics with the work of linguists.[16] Between the ages of 3 and 5, children start to focus on the language that they speak as an object of cognitive attention – a problem space in its own right – and demonstrate an ability to reflect metalinguistically on aspects of spoken language.[17] In their interactions with the linguistic environment, children engage in the construction of "consciously accessible theories about how language functions as a system." [18] In this constructive process, children are making generalizations, forming theories, testing the theories, and reformulating or refuting their theories.

This scientific method which children employ in their conscious exploration of the grammatical structure of spoken language is also applied to written language when children start to read and write. Emilia Ferreiro's work on children's conceptual development of written language shows that "children have ideas, and indeed, hypotheses and theories, which they continually test against the many examples of written text they encounter in their environment and against the information they receive from others." [19]

The linguistic explorations of children contribute significantly to their understanding of linguistic phenomena and to their general cognitive growth.[20] Similar to how children can use intuitions about body movement to open an intellectual path to understanding formal mathematical concepts, children can use intuitions about their language to open an intellectual path to understanding language as a formal system. Therefore, children's intuitive explorations of language – their processes of problem solving and theory building in the domain of language – should be encouraged and supported to continue, even as they work towards gaining understanding of formal linguistic concepts.

---

[16] Karmiloff-Smith92.
[17] Ibid.
[18] Ibid., pg. 51.
[19] Ferreiro82, pg. v.
[20] Karmiloff-Smith79.

Computer programming as an expressive medium for linguistic explorations ★

I believe that a computer programming environment can be used to effectively encourage and support children's intuitive explorations in the linguistic domain, in a manner similar to how Papert's Turtle geometry is used to encourage and support children's intuitive explorations in the mathematical domain. Of course, it is not necessary to work with computers in order to acquire good strategies for thinking of language in formal terms – I myself started on my path of thinking about language as a formal system without the aid of a computer. However, I have personally found that it is easier to think about language as a formal system when linguistic inquiry is pursued in a computational environment.

During my first year of graduate school, I was recommended to read a book entitled *Exploring Language with Logo* by Ernest Paul Goldenberg and Wallace Feurzeig. With the aid of this book, I was able to use the Logo programming environment as a medium for modeling aspects of language. In the computational environment, I worked to articulate my intuitions and investigate my working theories about various linguistic phenomena through the creative process of making tangible artifacts, in the form of computer programs. In essence, the computer programs that I made are *my* formal representations of those linguistic phenomena. Basically, the task of articulating elements of my intuitive theories for the purpose of constructing a systematic representation in the computer forced me to *externalize* and *formalize* my intuitions. In order to make a computer program that systematically worked the way I wanted it to, I had to confront and get rid of any obscurities and inconsistencies in how I thought about the particular aspect of language I was trying to model. In this way, the purpose and the descriptive power of formal grammatical concepts became something that I could personally relate to and understand.

For my thesis, I worked to contribute to children's appropriation of formal ways of thinking about language. To this end, I used the Logo programming environment as an expressive medium, through which both the children and I could use our linguistic intuitions and strategies in problem solving and theory building, to explore aspects of linguistic structures. Through the Logo programming environment, the children and I used the computer as an "object-to-think-with" and a "tool-to-construct-with", for the purpose of working towards externalizing and formalizing our intuitions about a particular aspect of language, and using our articulated intuitive theories to construct a formal representation (in the form of a computer program) of that particular aspect. In so doing, we were able to make connections with some formal concepts in the field of linguistics.

STRATEGIES FOR INQUIRY

Mobilize children's linguistic intuitions ★

The overarching goal of my thesis work is to motivate children to use their linguistic intuitions as a means towards gaining understanding of formal linguistic concepts and seeing language as a formal system. As a first step towards this goal, I wrote a computer program in which a turtle character passes its judgment on input words. The turtle character accepts some words while rejecting others, operating by a consistent linguistic criteria. This computer program was presented to children as a puzzle for them to solve. Children were asked to figure out the turtle character's criteria for accepting words.

The computer program served the purpose of being a stimulus to mobilize children's linguistic intuitions. After telling children what they had to figure out, I became a silent observer. I did not provide guidance on strategies to use and gave no helpful hints on how to interpret the turtle character's judgments. Basically, I let children direct their own inquiry and I let them grapple with difficulties they encountered in the program. When left to their own resources, children had to reason from within and appeal to their intuitions about the English language. This was a setting in which children could freely exercise their linguistic intuitions. Children's interaction with my computer program, their actions and verbalizations, provided for me a firsthand verification of my belief that school-aged children are capable of using their intuitions about language to effectively explore a linguistic problem space.

Use children's intuitive understanding to facilitate understanding of formal linguistic concepts ★

Children were able to arrive at an intuitive understanding of the turtle character's criteria for accepting and rejecting words. They explained that the turtle character accepts words that "look like" English. The children were able to use their intuitions about what English words look like to predict whether the turtle character would accept or reject a particular input word. To get into *how* the turtle character "knows" what English words should look like (and for that matter, how an English speaker knows what English words should look like), I guided children through the process of making the computer program.

Taking children through my programming process provided a meaningful context for introducing children to formal concepts like the sound pattern of English, letter-to-sound correspondences in English, the syllable and its division into onset, vowel, and coda positions, and constraints on how the sounds may combine in the English syllable. Understanding these formal concepts, and how they fit together to provide an account of the intuitions English speakers have about what makes a word look (and sound) like English, was a necessary ingredient in my construction of the computer program. Without an understanding of these formal concepts, I would not have been able to get my computer program to work.

I believe that children's understanding of the formal linguistic concepts that were used in my computer program was facilitated in two ways. First, children already understood how the computer program worked on an intuitive level. They could use their intuitions about what English words may look like to explain the turtle character's judgments. Secondly, the formal concepts were introduced to children in the context of the computer program. The computer program was something that was fun and interesting to play with, and because of this children were motivated to learn how it was made. As children worked to understand this, they needed to make an effort at understanding the formal concepts which were used to make the computer program.

Use the computer as a "play-partner" ★

My intention in creating the computer program was not to educate children on "facts" they should know about the English sound/spelling pattern. In other words, I did not use the computer as a "teaching machine." Rather, I used the computer as a "play-partner." And what I designed this play-partner to do is to help children explore aspects of formal linguistics which I myself have meaningfully appropriated through appealing to my linguistic intuitions.

My computer program about the English sound/spelling pattern was not an activity that I "prescribed" to children. What I did was "share" with the children the product of my own intellectual journey. The computer program is an articulated form of the processes I went through in trying to make sense of the English sound/spelling pattern phenomenon, processes through which I came to grasp some formal concepts in linguistics. I am confident that children will gain insights from the process of making sense of my programs only because I know I gained a lot from the process of getting my programs to make sense!

Involve children in programming the computer ★

I used the programming language Logo to implement my computer program. Logo provides an intuitive, easy-entry route into understanding how to program the computer. Compared to other programming languages, Logo commands are easier for children to use and understand. The purpose of each Logo command in the context of a particular procedure is also easily understood.

I wanted children to be actively involved in the evaluation and modification of my computer program. For this reason, in addition to inviting children to play with my program, I walked them through the process of making the program. I showed them the page where I wrote all the procedures that make up the program. In plain English, I explained the role of each command in a procedure, and the role of each procedure in the program. I focused on the organization of the program, rather than the exact syntax of Logo programming. I also pointed out the formal concepts in linguistics which I was able to use to construct a coherent program that constituted a model of the English sound/spelling phenomenon.

After I finished with my walkthrough of the programming process, I encouraged children to make modifications to my program, to pursue their ideas on how things should work differently. I also invited children to use my program as a template on which they could construct their own computer programs.

Act as children's observer, catalyst, and collaborator ★

The approach I took in working with children was adopted from Mitchel Resnick's description of how he worked with participants in his doctoral research project. Resnick simultaneously played three distinct but intertwined roles: he was the children's "observer", "catalyst", and "collaborator." [21]

In the context of my thesis, I first observed how children went about figuring out the turtle character's criteria for accepting words. When children had arrived at an intuitive understanding of the turtle character's criteria for accepting and rejecting words, I acted as a catalyst for the development of their concrete understanding of formal linguistic concepts. I asked children questions about what they were doing and what they were thinking about, and I challenged them to back up their hypotheses and explanations with concrete examples.

---

[21] Resnick92.

When I talked to children about the process I went through in creating my computer program, I acted as both an observer and a catalyst. I noted their reactions and the questions that they asked. I relayed my understanding of formal concepts and shared with them how I had arrived at my understanding of the formal through the use of my intuitions about language. I also asked them to critically evaluate my programs.

When children wanted to make changes to my program or use my program as a template for building their own computer programs, I acted mainly as a collaborator, but also as a catalyst and an observer. I worked with children on defining what they would like to do and provided help with the programming. At the same time, I asked children to explain their thoughts and ideas, while I shared my own ideas, and observed their thought processes and work processes.

THE STUDY

Participants' background ★

Twenty-one children from 6 to 12 years old (1st through 7th grade) participated in the study – eight girls and thirteen boys. There were two 6-year-olds (both girls), two 7-year-olds (both boys), two 8-year-olds (one girl and one boy), six 9-year-olds (four girls and two boys), six 10-year-olds (one girl and five boys), two 11-year-olds (both boys), and one 12-year-old (a boy).

All of the children speak English: two of the twenty-one are non-native speakers, the rest are native English speakers. Two of the children are home-schooled; the rest attend public schools in the greater Boston area. All of the children come from middle-income and high-income families. Most of the parents are working professionals and some are graduate students at MIT.[22]

All of the participants have at least one computer at home – one boy told me his family has eleven computers at home! The children mainly use their computers for typing up homework assignments, surfing the Internet, and playing CD-ROM games. A few children use their computers for drawing and writing stories. Four of the children have prior programming experience: they have very limited experience with Logo. All of the children stated that they like using computers.

I am aware that this was a very selected group of children indeed, which happened to be an advantage in an exploratory study like this one. The rationale is that, in effect, it is good to have a very selected group of "informed users" in the early phase of my participatory design adventure. After this initial phase of design, I do plan to broaden the population for further "tool-testing" and user appropriation studies.

---

[22] In recruiting children, I accepted every child whose parents contacted me in response to my advertisement via the MIT Media Laboratory lab-wide mailing list and flyers posted around the MIT campus.

Their ideas about "spelling", "understanding grammar", and "thinking about language" ★

Before the first session, I asked the children to fill out a questionnaire.[23] It was a way for me to get the know the children. In the questionnaire, the majority of the children expressed that they like reading and writing, but also expressed that they are not "good at spelling", do not "understand grammar", and do not like to "think about language."

In asking children to explain their responses, I found that more than half of the children think that they are "horrible" spellers. They attribute this to having "a bad memory." The majority of the children have regular spelling tests, for which they have to study for by memorizing a list of words. Most children said that they don't do well on spelling tests. Based on what this group of children told me, it seems that the entire enterprise of spelling tests is focused on categorizing children as "a good speller" or "a bad speller" than on helping children learn how the correct spelling of words. Children get points for words spelled correctly. Words that are spelled incorrectly are circled or in some other way marked as wrong, but the correct spelling is not provided. Nor is there any direction provided on how children can go about learning the correct spelling.

Children also stated that they "don't know what grammar is." One 11-year-old boy explained, "Some of the words [teachers] use [in school] I don't understand ... It's hard to know what [the words] mean ... It's like foreign language ... And when I don't know the meaning, it's hard [for me] to understand." The other children's responses echoed the same sentiments towards grammar.

When I asked the children about why they didn't like "thinking about language." It turned out that most of them didn't even know what I meant by "thinking about language." They nonetheless just assumed that it couldn't be something that they would enjoy. Instead of giving children "the answer" to what I think "thinking about language" means, I worked with the children to come up with examples of "thinking about language." One 9-year-old girl came up with "wondering about people's accents" as an example of "thinking about language." One 7-year-old boy came up with "making up meaning to words" – which is actually something he often does for fun. A 12-year-old boy offered that an example of "thinking about language" could be "wondering about why some words are spelled the way they are ... some of those words are such a pain to memorize for the spelling test."

---

[23] The questionnaire can be found in the appendix.

Period of participation and sequence of activities ★

For a period of eight weeks, starting on February 17 and ending on April 20, 1999,[24] children participants came to the MIT Media Laboratory in groups of three to five, for two to three hours each week. There were five groups of children total.

The group which came in on Mondays from 3 to 5pm consisted of five children: KS (girl: age 8, grade 3), TN (boy: age 8, grade 5), ID (boy: age 9, grade 3), UT (girl: age 10, grade 5), and WA (boy: age 11, grade 6).[25]

The group which came in on Tuesdays from 3 to 5pm consisted of three children: AL (girl: age 6, grade 1), DE (girl: age 6, grade 1), and CA (boy: age 7, grade 2).

The group which came in on Wednesdays from 2:30 to 5:30pm consisted of three children: GA (girl: age 9, grade 3), HI (girl: age 9, grade 3), and LA (girl: age 9, grade 4).

The group which came in on Saturday mornings from 10am to 12noon consisted of five children: PW (boy: age 9, grade 4), NE (boy: age 10, grade 4), MP (boy: age 10, grade 4), OH (boy: age 10, grade 4), and RI (boy: age 10, grade 4).

The group which came in on Saturday afternoons from 1:30 to 4pm consisted of five children: FM (boy: age 7, grade 2), ME (girl: age 9, grade 4), SC (boy: age 10, grade 4), VT (boy: age 11, grade 6), and AK (boy: age 12, grade 7).

At my first meeting with each group of children, I presented to them the program I made about the English spelling pattern. Children played at figuring out what type of words the turtle character likes and what type of words it does not like. During the second week, I gave children a walkthrough of how I made the program and asked children to become program evaluators and editors. During the third, fourth, and fifth weeks, children used my original program as a template from which they made spelling pattern program for languages that they know (Malay and Polish) or for languages that they invented. For the sixth, seventh, and eighth weeks, children were focused on coming up with their own ideas for making computer programs about language.

---

[24] We did not meet during the week of March 21, 1999 because of spring vacation.
[25] To ensure anonymity, the initials used here are not the children's actual initials.

PLAY SPACE

Spelling patterns: A world of possible words ★

The spelling of a language, with all of its complexities, is quite a lawful system. This is because each language has its own distinctive sound pattern, what linguists call its phonology. The phonology of a language consists of an inventory of sounds ("phonemes") that exist in the language and constraints on how the sounds may be combined to form "natural-sounding words" in the language.[26] To illustrate this point, consider the following set of words: *plaft*, *thax*, *hlad*, *mgla*, *tesn*, *flutch*, and *srim*. Obviously, none of them are actual words in English; however, English-speakers intuitively distinguish *plaft*, *thax*, and *flutch* as possible words in the English language, whereas the remaining ones are not – and could never be – possible words in English.[27] Our ability as speakers of English to distinguish between possible and impossible nonsense words results from our intuitions about which sound sequences are permitted in each part of the English syllable.[28] The syllable consists of three parts: a vowel, which is an obligatory constituent, an onset – a consonant or consonant cluster in syllable-initial position, and a coda – a consonant or consonant cluster in syllable-final position.[29] English speakers intuitively rule out words like *hlad*, *mgla*, and *srim* as possible English words because */hl-/*, */mgl-/*, and */sr-/* are not permitted sound sequences in the onset position. Along the same line, *tesn* is ruled out because */-sn/* is not a permitted sound sequence in the coda position in English.

The phonology of a language is generally reflected in its representation in print, although the degree of correspondence varies from language to language. For English, the relationship between letters and sounds is not an exact one to one correspondence.[30] Still, there are regularities in English spelling that reflect the underlying sound pattern, which I was able to capitalize on for the purpose of introducing to children the phenomenon of linguistic constraints governing the syllable.

---

[26] Chomsky68.
[27] Pinker94.
[28] Chomsky68.
[29] Kenstowicz94.
[30] Ibid.

Making a computer program about the English spelling pattern ★

Working from my understanding of the model of spelling patterns as arising from constraints governing the syllable, I wrote a Logo program[31] that takes an input word and parses it into syllables and evaluates each element in the syllable against defined sets of allowed vowels, onsets, and codas. The first procedure of the program defines allowed sets of vowels, consonants, onsets, and codas.

```
to startup
setvowels
       [a e i o u
        ae ai ao au
        ea ee ei eo eu
        ia ie io iu
        oa oe oi oo ou
        ua ue ui uo]
setconsonants
       [b c d f g h j k l m n p q r s t v w x y z]
setonsets
       [b c d f g h j k l m n p qu r s t v w x y z
        bl br
        ch cl cr
        dr dw
        fl fr
        gh gl gr gw
        kl kr kw
        ph pl pr
        sc sh sk sl sm sn sp squ st sw
        th tr tw
        wh wr
        chr
        scl scr shl shr skl skr spl spr str
        thr thw]
setcodas
       [b d f g k l m n p r s t v w x y z
        dd ff gg ll rr ss tt zz
        ch ck
        ft
        ld lf lk lm lp lt
        mb mn mp
        nd ng nk nt
        ph pt
        rb rd rf rk rl rm rn rp rt rv
        sh sk sm sp st
        th
        wk wl
        bs ds gs ks ls ms ns ps rs ts ws ys]
end
```

[31] The entire set of procedures for this program can be found in the appendix.

I came up with these sets of allowed elements by relying on my own intuitions about how words are spelled in English. The last line of elements in **setcodas** (from **bs** to **ys**) were added to the set on my second look so that the program would accept plural words like *cobs*, *beds*, *dogs*, and *keys*.

My intention was not to come up with a perfect account of English spelling in my initial sets of allowed elements. I only wanted the sets to be representative of the typical combinations that exist in English words, so that children can find some kind of logic in my program and pick up on the sound/spelling pattern.

A crucial part of my experimental design was my intention to actively involve children in the evaluation and modification of my programs. I wanted children to play an integral part in making decisions about what elements to add to and subtract from my initial list to make the program a more accurate and complex model of the English language. I also wanted to leave it open for children to debug my program and add their own layers of complexity to the program.

The second procedure is a superprocedure that consists of procedures for parsing an input word into syllables and procedures for evaluating each syllable of the input word against the allowed sets of consonants, vowels, onsets, and codas. When a word is typed into the input textbox, the program runs the second procedure which first looks for a vowel or vowel cluster, as defined in **setvowels**. If there is an allowed vowel or vowel cluster in the word, the letter or letters in front of the vowel or vowel cluster are reported to a textbox called "onset" and are checked against the allowed set of onsets. The letter or letters behind the vowel or vowel cluster are reported to a textbox called "coda" and are checked against the allowed set of codas.

An input word is determined to be a possible word in English when no disallowed elements have been found (Figure 1).

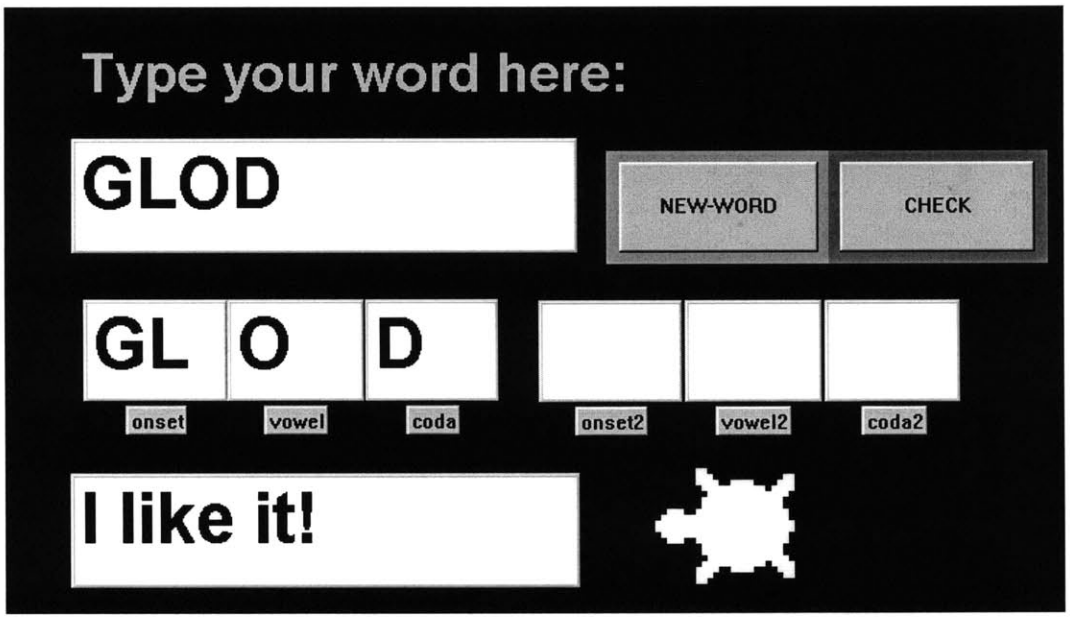


Figure 1 A nonsense word that complies with the English spelling pattern

Words that have a disallowed element in the onset position (Figure 2), the vowel position (Figure 3), or the coda position (Figure 4) are judged to be unacceptable.

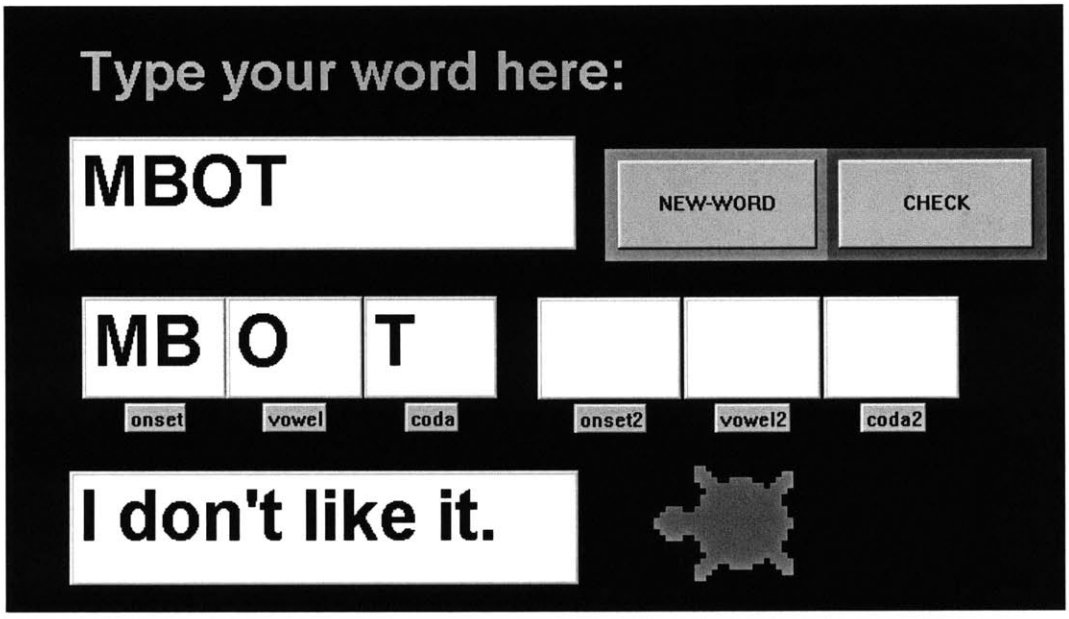


Figure 2 A nonsense word with a disallowed onset

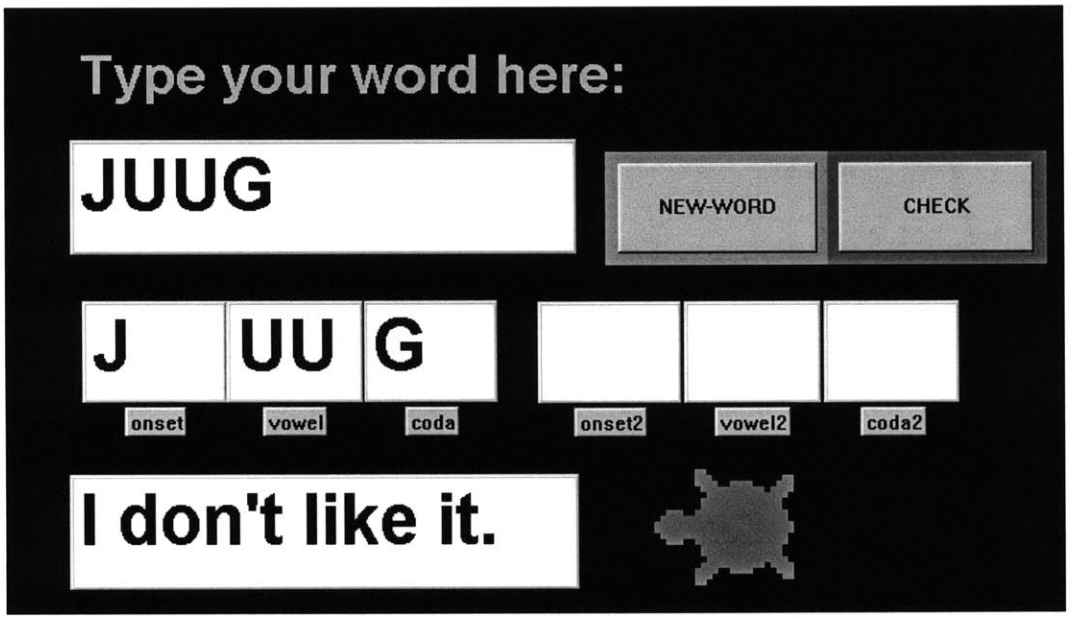


Figure 3 A nonsense word with a disallowed vowel

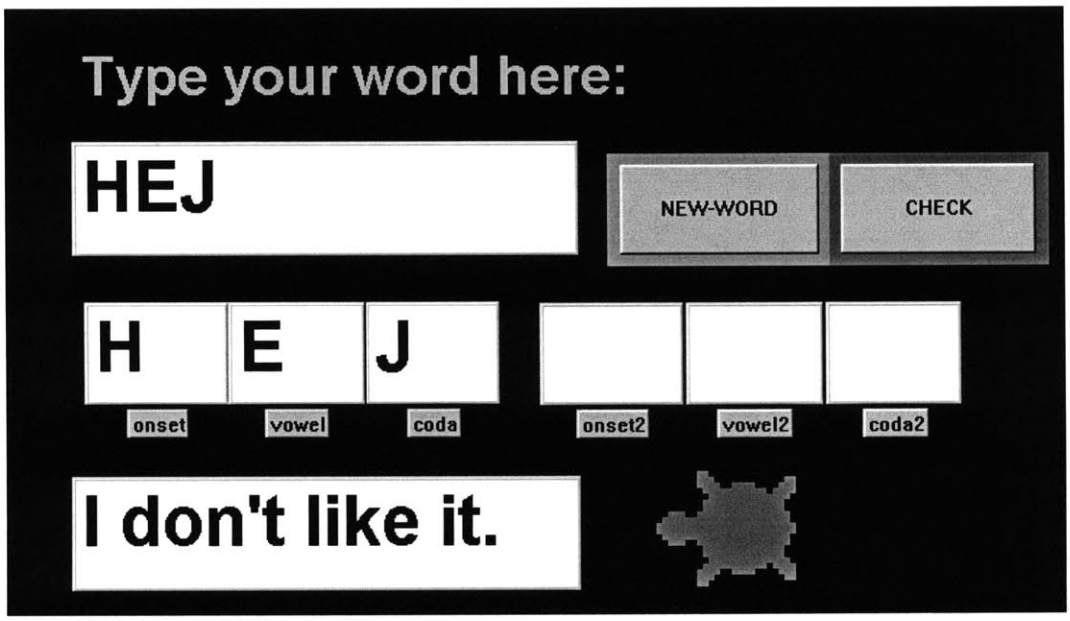


Figure 4 A nonsense word with a disallowed coda

The input word has to minimally have an onset and a vowel (Figure 5) or a vowel and a coda (Figure 6).

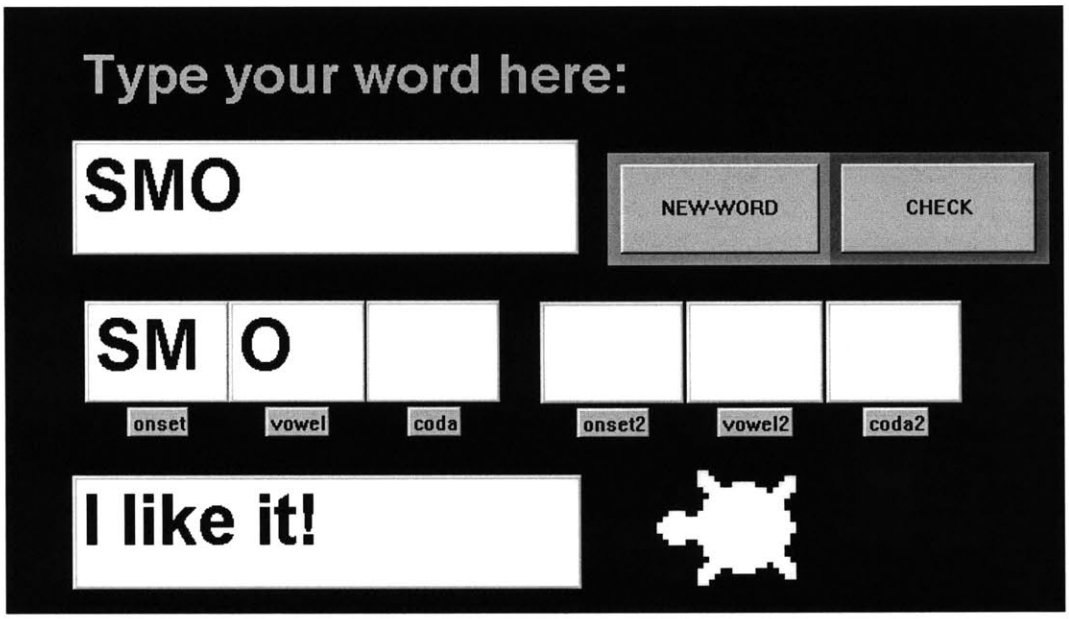


Figure 5 A nonsense word without a coda

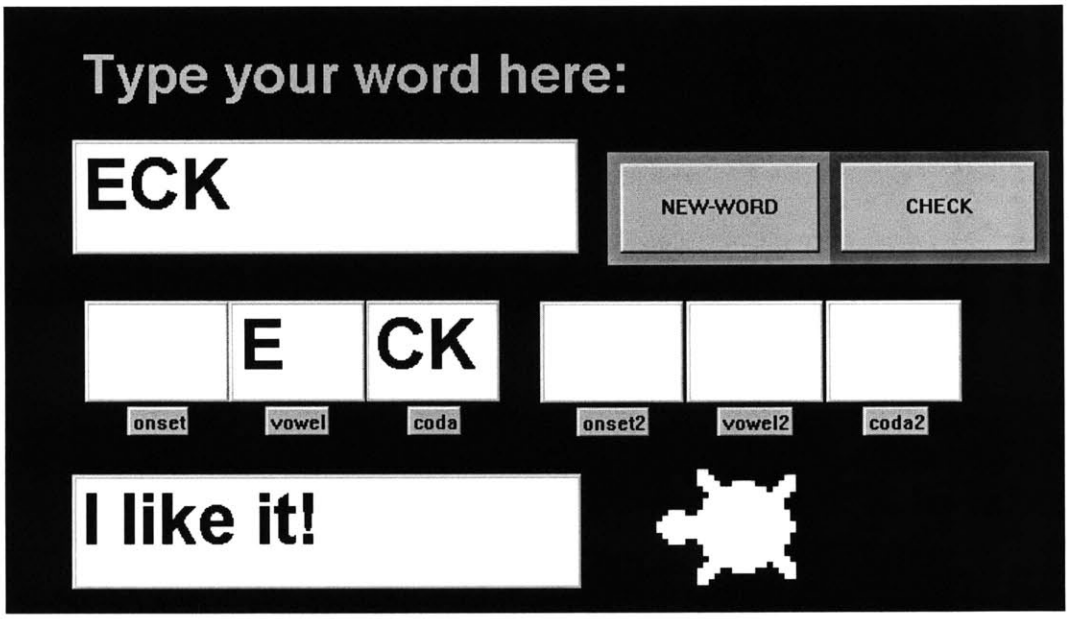


Figure 6 A nonsense word without an onset

The program's judgments are communicated through an artificial play partner: the turtle. After each proposition made by children, the turtle says "I like it!" to words – actual or nonsense – that fit the English spelling pattern, as defined in the allowed sets of elements. It says "I don't like it", and changes its color from green to red, when it encounters words that do not fit the English spelling pattern.

For input words with two syllables, the program will encounter another vowel or vowel cluster in the "coda" textbox after the first set of parsing and evaluating procedures are finished. The presence of a vowel or vowel cluster in the "coda" textbox triggers a second set of parsing and evaluating procedures. The second set of procedures takes the second vowel or vowel cluster and reports it to a textbox called "vowel2." The vowel or vowel cluster is then checked against the allowed set of vowels. The letter in front of the second vowel or vowel cluster is then moved to a textbox called "onset2" and checked against the allowed set of onsets. The letter or letters behind the second vowel or vowel cluster are reported to a textbox called "coda2" and are checked against the allowed set of codas.[32] The program determines the input word to be a possible word in English when no disallowed elements have been found in either syllable (Figure 7).

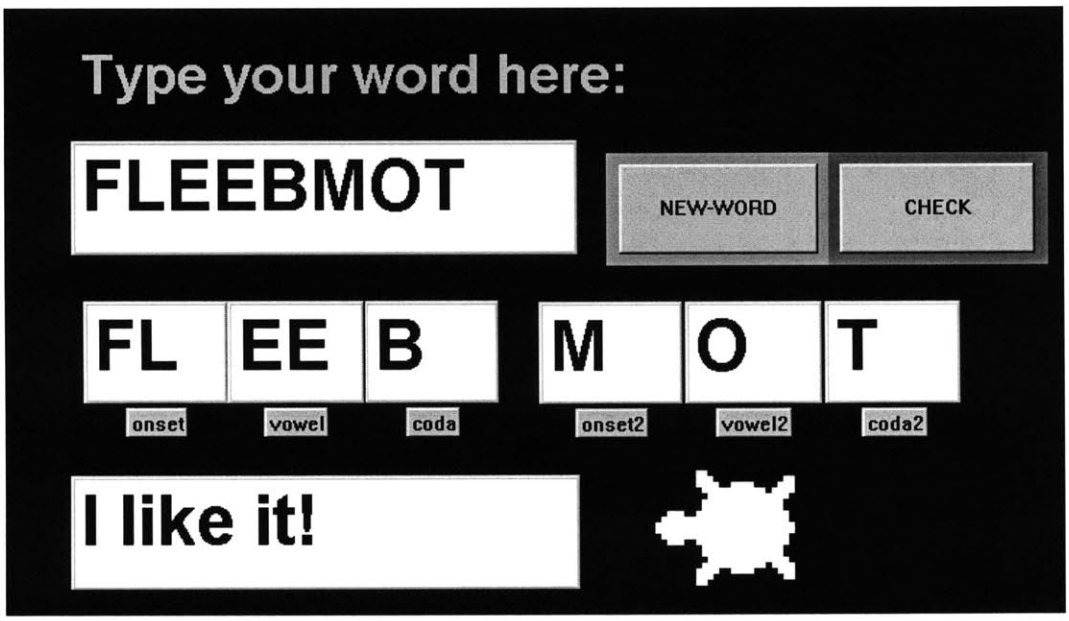


Figure 7 A two-syllable nonsense word

[32] At this point, the program cannot handle words with more than two syllables.

To ensure that no input words will have more than two syllables, I programmed in a restriction to limit the length of the input word to four letters or less – the turtle will say "Too long!" to words more than four letters long (Figure 8).

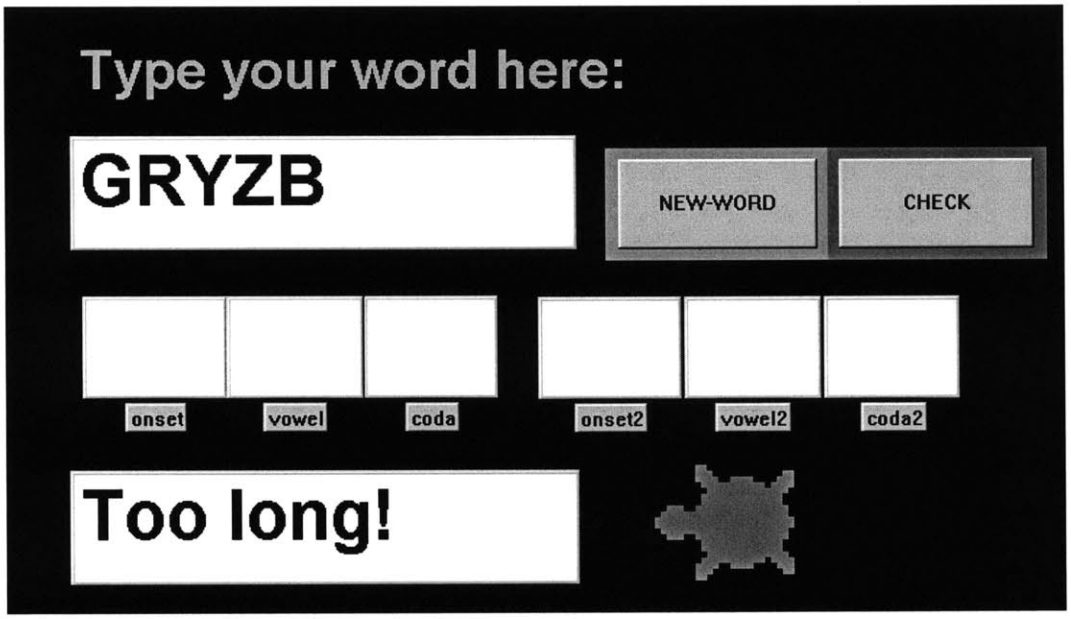


Figure 8 A nonsense word that is "Too long!"

Presenting my English spelling pattern program to children ★

WEEK ONE: At my first meeting with each group of children, I presented to them the program I made about the English spelling pattern. I didn't tell them, of course, that it was a program about the English spelling pattern – that was one of the things they had to figure out.

After we finished introducing ourselves and the parents left, I gave the children some snacks and asked the group to huddle around a table with two computers set up side by side. I clicked open my program, which looked like this on the screen (Figure 9):

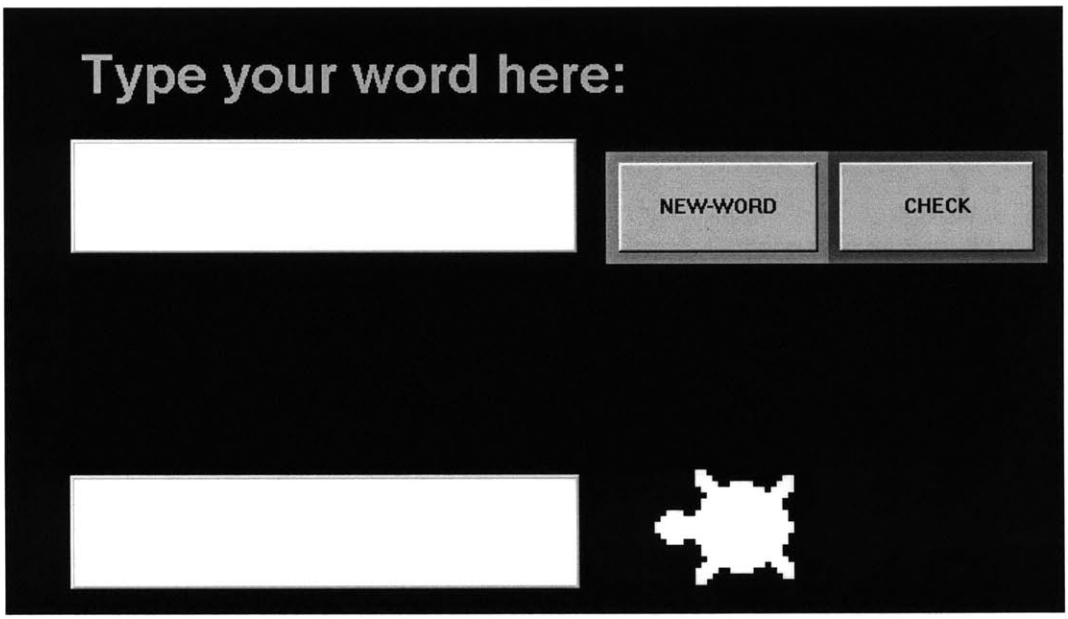


Figure 9 My English spelling pattern program as presented to children

I told each group of children that I had spent some time working on this program, that the program is still "sketchy", and that I would like some help in "testing out" my program.

To play with the program, I explained, you type in a word in the textbox at the top of the screen, where it says, "Type your word here." Then you use the mouse to click on the button that says "CHECK." The turtle will then tell you in the bottom textbox whether it likes the word or not. When you want to type in a new word, click on the button that says "NEW-WORD", the top textbox will then become empty and you can type in your new word. The goal is to figure out what type of words the turtle likes and what type of words it does not like.

I gave no other information beyond this. As shown in Figure 9, children could only see the input textbox and the message textbox; therefore, they did not see how the program parsed the input words. Children, playing in groups of two or three, typed in words, noted the turtle's response to each word, and worked at figuring out the basis of the turtle's judgments.

WHAT DID THE CHILDREN DO ?

Children making sense of my English spelling pattern program ★

All children started out with a meaning-based strategy. In trying to come up with a word the turtle does not like, NE, OH, and MP (all age 10, grade 4) asked themselves "What's a turtle's enemy? What eats a turtle?" They then started entering words like HAWK, BIRD, DOGS, CARS, and HOGS. VT (age 11, grade 6) entered *TURTLE[33] as his first word, to which the turtle replied "Too long!" – in response, VT exclaimed, "It doesn't like itself!" CA (age 7, grade 2) typed in MAN and BALL in his attempt to figure out what words the turtle likes. AL (age 6, grade 1) tried the words *HAPPY, PET, HUGS, and KITE in her attempt to figure out what words the turtle likes.

Children quickly arrived at two conclusions: the turtle does not like words longer than four letters and the turtle likes actual English words. But the children were not satisfied, they pressed on to test the robustness of these ideas.

OH (age 10, grade 4) wanted some negative evidence to confirm his conclusion that the turtle likes real words. He told his friends: "Let's try four letters that don't make a word." OH then typed in *TTTT, then *JHYG. Both times, OH voiced his prediction that the turtle was not going to like it. FM (age 7, grade 2), working in another group, had the same idea as OH: "I'm going to try words that it won't like." With this declaration, FM tried *GHJ, *FGA, and *AZX. LA (age 9, grade 4) also independently proposed the idea of "making a word that the turtle won't like." She tried *MOBD, *URSW, and *QWEN. As LA typed in *QWEN, her friend HI (age 9, grade 3) chimed in "The turtle is not going to like it because it's always *qu* in English."

Across the board, the children were very good at forming hypotheses based on what they observed and coming up with words to test their hypotheses. They were eager to share their ideas and comment on each other's thoughts. For instance, when NE (age 10, grade 4) asserted that "when a vowel is the first letter, the turtle breaks it up to try to make a real word" (typing in OPOP to demonstrate his point), his friend MP (age 10, grade 4) immediately jumped in and typed JIJI, TETE, and PERE to

[33] The symbol * denotes words that the turtle does not like because of the length or because of disallowed elements.

prove him wrong. NE quickly responded by revising his hypothesis, taking out the condition that the first letter be a vowel. In another group, when VT (age 11, grade 6) proposed that the turtle likes words with at least one letter from the word "turtle" in it, except for T, FM (age 7, grade 2) protested, "But it likes TIG!"

About thirty minutes into the first session, I suggested to the children that they play a game of Scrabble[34] with the help of the turtle. I asked them to play with disregard to the normal rules of the game and to play by the following two rules: only words that the turtle likes are allowed and actual English words are not allowed. It was interesting to observe the children play the Scrabble game. While the children maintained that they were still unsure of how the turtle judges words, they demonstrated a high degree of certainty in selecting words to play the game. They often commented on their choice of words and the other children's choice of words, saying things like "I know that's going to work!" or "I think the turtle is going to like it!"

On average, the children started to develop ideas about what words the turtle likes and what words it does not like after about forty minutes of playing with the program. Some children (mainly children older than age 8) were able to explain the reasoning behind their theories with a higher degree of explicitness than other children. Within an hour of playing with the program, the children – learning from their own and others' actions and verbalizations on word selection – came to similar conclusions about how the program worked.

OH (age 10, grade 4) concluded that the turtle likes "proper words that are real words" and "proper words that are not real words." VT (age 11, grade 6) concluded, "you can make up words but laws of English still count." ME (age 9, grade 4) explained that "the turtle likes words that could be right." TN (age 8, grade 5) said that words the turtle does not like are "words that you can't pronounce." CA (age 7, grade 2) explained how he could tell which words the turtle would not like: "I look at it ... and I don't know this word ... so the turtle is not going to like it." LA (age 9, grade 4) commented that words the turtle likes "don't have to be real words ... they only have to *look* like real words ... they have to have the right letters." HI (age 9, grade 3) agreed with her friend: "The words can be words that aren't real English but look like English ... because of the spelling."

---

[34] "In Scrabble, players form interlocking words cross-word fashion on the board using letter tiles. Words not found in the dictionary are not permitted." (from Scrabble® Game Rules)

From the activities of the first week, it became apparent that the children felt intellectually challenged by the program, as embodied by the turtle. They enjoyed the challenge and found it satisfying when they finally figured the program out. MP (age 10, grade 4) proudly reported to his mother: "The turtle was mean, but in the end we figured him out." WA (age 11, grade 6) explained that the turtle "forced [him] to think a lot." UT (age 10, grade 5) said that she liked playing with the turtle because "he had a mind of his own."

Children evaluating my English spelling pattern program and making modifications ★

WEEK TWO: After giving children a walkthrough of how I made the program, I asked them to play with the program some more – this time with a focus on how the program parses the input words (Figure 10). The children became program evaluators and editors. Through discussions with me, they/we debugged the program to fit their preferences.

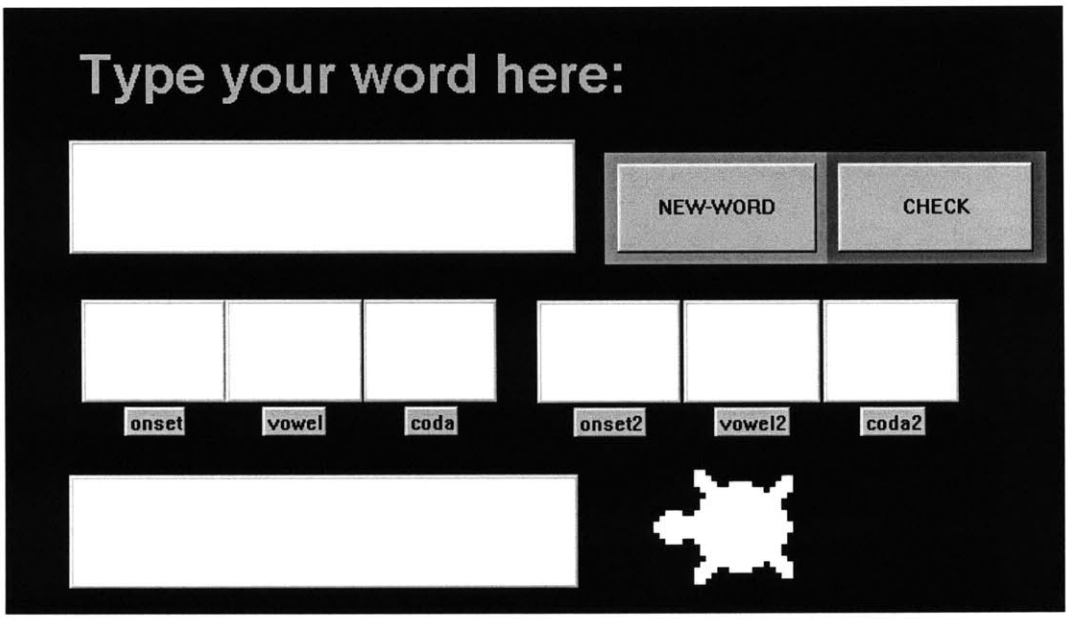


Figure 10 My program with all of the textboxes visible

The Monday group (KS: age 8, grade 3; ID: age 9, grade 3; TN: age 8, grade 5; UT: age 10, grade 5; WA: age 11, grade 6) typed in the word MANY and discovered a bug. UT exclaimed, "That's a real word – why doesn't the turtle like it?" The group spent some time poring over the allowed sets of elements and came up with a tentative solution. ID suggested that in MANY, *y* is a vowel: "It sounds like an *e*." So *y* was added to the allowed set of vowels. By doing this, the children ran into another bug – *y* was now defined as both a consonant and a vowel, and that rendered the program nonfunctional. While this is how most English speakers think of *y* – KS even recited for the group something she remembers from her English class: "Vowels are *a, e, i, o, u,* sometimes *y* and *w*" – the children finally concluded that it was "too confusing" for the computer to "think" of *y* as both a consonant and a vowel (because "the computer is too stupid"). The group then came up with an ingenious solution. They settled on defining *y* as a consonant, and added **by dy fy gy hy ky ly my ny py ry sy vy** and **zy** as new elements in **setcodas**. The children felt that this was a satisfactory solution, even though it does not reflect how English speakers think.

The Wednesday group (GA: grade 3, age 9; HI: grade 3, age 9; LA: grade 4, age 9) also came across a difference between their way of processing English and the computer's way. GA noted that the computer "breaks up" some words differently than she would. She explained that she thinks of words like MINE, CUTE, and LIKE as having one syllable; the computer, on the other hand, breaks them up into two syllables (MI-NE, CU-TE, and LI-KE). To explain, I took the group back to the page where I wrote all the procedures that make up the program. I explained that the procedure which parses the input word parses the word into two syllables if there are two separated vowels or vowel clusters in the word. This is the case with MINE, CUTE, and LIKE. However, I explained, there is a phenomenon in English called "the silent *e*." (GA chimed in, "Oh, I think I've heard of that!") In a sense, the *e* at the end of most English words are not pronounced: we don't pronounce *mine* "mee-nay", or *cute* "cue-tay", or *like* "lee-kay." Basically, the computer breaks up these words that have the silent *e* differently than we would because the computer doesn't "know" about the silent *e* phenomenon in English. I suggested to the group that I could help them write a procedure for addressing words ending in *e*. However, the group decided not to pursue the idea. GA and her friends concluded that they wanted to keep the program as it was because they didn't see anything wrong with "the computer's way." They only thought that it was interesting, "because we are so different, computers and humans."

The Saturday afternoon group (FM: grade 2, age 7; ME: grade 4, age 9; SC: grade 4, age 10; VT: grade 6, age 11; AK grade 7, age 12) wanted to change the word length parameter to accommodate six-letter words. I showed the group where they could make that change in the program's procedures. After they made the change and

started to play with the program, they discovered that with more letters, the possible number of syllables increased. The program was now unable to evaluate some of the input words – namely, words with three syllables. The group recognized this and wanted to fix the program. Slowly, I guided them through the process of adding procedures for parsing and evaluating a third syllable. We used the existing parsing and evaluating procedures as templates for our new procedures, and I think the group gained a better understanding of how the program was made from this experience.

Children's intuitions about the letter *y*, the silent *e*, and the relationship between word length and syllable number enabled them to meaningfully critique the limitations of the model of the English spelling pattern embodied in my program. Through working to articulate and implement their intuitions, children demonstrated to me a developing understanding of some of the formal concepts embedded in my English spelling pattern program. These concepts included letter-to-sound correspondences, syllable structure, and syllable boundaries.

Children modifying my program for Malay and Polish spelling patterns ★

WEEKS THREE AND FOUR: There were children in the Tuesday and Wednesday groups who speak another language as their native tongue, or have family members who speak another language as their native tongue. These children wanted to make a program that models the spelling pattern of their language.[35] To get them started, I asked them to compile a list of words that they felt were representative of the consonants, vowels, consonant combinations, and vowel combinations present in their language.[36] I then helped them determine allowed sets of elements in their language, using the words from their list.

CA (age 7, grade 2) and AL (age 6, grade 1), who came in on Tuesdays, speak Malay as their first language. To make a program that is able to evaluate whether or not an input word conforms to the spelling pattern of Malay, I helped CA and AL replace the defined sets of allowed elements in my program with the sets of allowed elements they defined for Malay.

---

[35] Malay and Polish both use the Roman alphabet for writing.

[36] The children's lists of Malay and Polish words can be found in the appendix.

These are the sets of allowed elements CA and AL defined for Malay:

```
to startup
setvowels
       [a e i o u
        aa ai au
        ei
        ia
        ua]
setconsonants
       [b c d f g h j k l m n p r s t w y]
setonsets
       [b c d f g h j k l m n p r s t w y
        kh]
setcodas
       [b c d f h k l m n s t
        ng]
end
```

CA and AL had a lot of fun sharing their program with the other children. Pride in their program and in their language really showed on their faces as they answered the group's questions about the Malay language and Malaysia. Most of the children, some of whom did not even know where Malaysia was, expressed surprise at the similarities between English and Malay spelling. KS (age 8, grade 3)[37] excitedly shared her revelation with the group: "Real words in Malay look like nonsense words in English!" The differences of Malay spelling were also appreciated. TN (age 8, grade 5) said to AL, "Wow! You have lots of double *a*'s in your language." Everybody, including myself, had a fantastic time learning vocabulary words in Malay.

I was very happy to see CA and AL be the center of attention – they have been in the United States for only one year and were quite insecure about their English. I was very much the same way when I first moved to the United States from Thailand. My English was not very good, and I felt alienated from my peer group because of the "strange" language that I spoke. From the beginning of their participation, CA and AL mainly kept to themselves. AL, in fact, did not talk much in English. She would only occasionally whisper things in Malay to CA. Their particular take on my program entirely changed the situation. Through constructing a representation of Malay, CA and AL came to realize how much they know intuitively, and they were able to flesh out their knowledge and share their expertise with the other children.

[37] For the third and fourth weeks, the Monday group and the Tuesday group both came in on Thursdays.

LA (age 9, grade 4), who came in on Wednesdays, has parents who speak Polish. Her parents often speak in Polish to each other so LA is familiar with the sounds of the language – she is also familiar with how it looks in writing. LA even knows some Polish words herself. To get started on making a Polish spelling program, LA asked her parents to write down some Polish words for her. She then looked over the words and started to make a list of vowels and vowel clusters, consonants and consonant clusters that appear before the vowel position, and consonants and consonant clusters that appear after the vowel position. When LA finished with the words her parents gave her, she wanted to look for more words because she felt that "some letters were still missing."

I helped LA search the Internet and we found a website that has translated Polish words and phrases for travelers with sound files that helped us learn the pronunciation of the words and phrases.[38] Hearing foreign sounds coming from the computer, GA and HI (both age 9, grade 3) became interested in what LA was working on. LA, GA, and HI wrote down some words from the website. And as I have shown her before, LA instructed her friends to divide each word into syllables by drawing a line at the syllable boundary, so the consonants and consonant clusters can be classified as onsets or codas. HI immediately protested: "But we don't know how to pronounce the words so we don't know how many syllables they are supposed to have ... like *proszę* -- is it supposed to be *prosz-e* or *pro-sze*?" LA gave her friend a stern look and said, "Just go with what you think!"

When they were done with parsing the words, LA, GA, and HI compared notes and came to a surprising discovery. GA and HI divided up the words almost identically, and how they divided up the words differed significantly from how LA divided up the words. *Dobrze* was *dob–rze* according to LA, but it was *do–brze* according to GA and HI; *dworzec* was *dwo–rzec* according to LA, but it was *dwor–zec* according to GA and HI; *matka* was *ma–tka* according to LA, but it was *mat–ka* according to GA and HI. I challenged the group to come up with an explanation. After a few minutes of looking at each other's papers with knitted brows, they did come up with an explanation. LA explained that GA and HI's "mistakes" were due to the fact that "all they know is English ... so they didn't know that there are combinations like *rz* and *tk*." Even though LA labeled her friends' judgments as "mistakes", she recognized that the ways in which GA and HI processed the Polish words were a valid solution, consistent with their intuitions as English speakers.

[38] http://www.travlang.com

These are the sets of allowed elements LA came up with for her Polish spelling program:

```
to startup
setvowels
       [a e i o u y
        ia ie io
        oa]
setconsonants
       [b c d g j k l m n p r s t w z]
setonsets
       [b c d g j k l m n p r s t z
        br
        cj cz
        dl dw dz
        gr
        kw
        pr
        rz
        sk sr st sz
        tk tr
        wt
        zl
        czt czw
        prz
        skl szk szp
        trz
        wrz wzg
        zb zt]
setcodas
       [b c d g j k l m n p r s t w z
        rt
        sc st
        zd]
end
```

Children making a spelling pattern program for an "invented" language ★

WEEKS THREE, FOUR, AND FIVE: I suggested to the children that they could come up with their own language – an invented language with a unique set of allowed vowels, consonants, consonant combinations, and vowel combinations. The majority of the children took to the idea of having their own "language." To go about doing this, they first had to decide which vowels and consonants would exist in their language and how the vowels and consonants would combine in their language. Then they had to replace the four sets of allowed elements (**setvowels**, **setconsonants**, **setonsets**, and **setcodas**) in my English spelling program with their own sets.

After typing in the sets of allowed elements for their invented language, most of the children spent a lot of time playing with their program, making lists of allowed and disallowed words and challenging other children in their group to figure out what vowels, consonants, and combinations existed in their language. The children also worked hard to personalize their programs, decorating the screen with drawings and giving a name to their invented language.

FM (age 7, grade 2) came up with a name for his invented language. "*Faderkan* is the name of my language," he told me. FM's father overheard us and leaned over to look at his son's program. He asked FM, "Can you spell the name of your language with the letters you have?" FM looked at his letters and combinations for a while, then finally answered his father, "*Faderkan* is the English name for my language, *Sravj* is the real name ... just like Japanese and 'Nihongo'!" FM decorated his project with characters from the "Teletubbies" television show, which I had helped him find on the Internet. I guided him through programming the characters to dance when they like the input word and to disappear "into their house" when they don't like the input word. FM's project (Figure 11) was a big hit with his group. The rest of the children wanted animated characters in their projects, too – and FM was more than happy to help them out with the programming.

Figure 11 FM's new language "Sravj"

I found that the majority of the children really liked the idea of having a language that they "came up with" themselves. KS (age 8, grade 3) said to me, "I like this a lot ... because I like how I get to make up words to play with."

Children explaining to others what they did on the computer ★

WEEK FIVE: At the beginning of the fifth session on Wednesday, one of the parents stopped by with her friend to see what the group was doing. GA, HI, (both age 9, grade 3) and LA (age 9, grade 4) had been working on making a program for their invented languages – *Ishmoo*, *Bobom*, and *Princess*. The group eagerly showed off their programs and excitedly talked about how they had made their programs. I was surprised at the level of explicitness found in their explanations. For instance, in her explanation of the terms "onset" and "coda", GA offered some illustrative examples: "Like *st*, it can be an onset or a coda ... It's an onset in *stop* and a coda in *best* ... But like *dumb*, the *mb* can only be in the back in English ... look, *mbat* wouldn't be a word in English."

I was inspired by what the Wednesday group did and wanted to find out how the other children would explain their programs to an outside audience. So I devised a task – I asked each child to write a letter to someone the same age as them. In the letter, I explained, they could either talk about the English spelling pattern program that I made or the invented language program that they made. I asked them to talk about what the program is about and how the program works in their letter. I stressed that they should try to be as clear as they can in their explanations, because the person they are writing to does not know anything about what we are doing.

Most of the children chose to write about my English spelling pattern program. In their letters, most children – like children in the Wednesday group – meaningfully used formal linguistic concepts (such as vowel and consonant combinations, and onset, vowel, and coda positions in the syllable) in their explanations of how the English spelling pattern program worked. These are excerpts from some of the more articulate letters:

TN (age 8, grade 5) wrote: "We were trying to find out what a computerized turtle likes. He likes nonsense words. It decides by, if he sees vowel combinations that are not allowed he won't like the word. If it is he will allow the word. The same thing will happen with consonant combinations."

NE (age 10, grade 4) wrote: "The turtle uses proper English you could type in any random words and it will tell you if it could become a proper English word. It figures out it's an English word by a command sheet, a command sheet is where you can say that *a*, *e*, *i*, *o*, *u*, are vowels and how many letters you can use."

UT (age 10, grade 5) wrote: "You have to take a word type it and press "Check" and it will say either "I don't like it" or "I like it". When he's red, he didn't like the word, when he's green he liked it. He likes English words, and nonsense words, that look like English words. Like DROE. He puts the letters into groups, "Onset, Vowel, or Coda"."

CHILDREN'S OWN IDEAS

Helping children make their own computer programs ★

WEEKS SIX, SEVEN, AND EIGHT: I was interested in getting children to pursue their own ideas in the design and implementation of computer programs. I wanted to find out what they would come up with, what ideas they would choose to pursue, and how they would go about implementing their ideas in a computer program. So I asked children to think about making their own computer programs about language. The only restriction I imposed was that the program had to be about language in some way. I told children that they didn't need to worry about knowing the commands for programming the computer because I would help them work through the programming. Just come up with an idea, I explained, then we can work together on making the program happen.

Most of the children were enthusiastic about working on their own programs, but they got stuck on coming up with *what* they wanted to work on. I tried to get the children past this block by continuing to talk to them, individually and in groups, asking them to think about things that interested them about language and what they might want to investigate and communicate about those things. I tried to get a brainstorming session going for each group, using a whiteboard and big pieces of paper.

I knew going in that trying to get children to work on their own programs at this point was very ambitious. However, I still wanted to put the ball in the children's court to see what would happen – after all, I would rather err on the side of overestimating than underestimating the children. It turned out that most of the children didn't come up with any ideas on what they wanted to do, and I just let it drop when they started to get frustrated and didn't want to think about it anymore. For these children who didn't know what they wanted to work on, I let them play freely and left it up to them to approach me whenever they wanted to focus again on working to make their own computer programs.

It turned out that only three of the twenty-one children came up with ideas for their own computer programs. However, I was very encouraged by what these children came up with, because it indicated to me that these children have made connections on some level with what we have been doing so far.

First of all, all three children focused on the linguistic form as an object of cognitive attention in their programs. Secondly, the programs they came up with were programs that generated *something*. And most importantly, their programs were about "doing something fun" with linguistic elements.

In these three fundamental ways, the children's programs resonated with the spelling patterns activities I had engaged them in, even though these programs were not about investigating a natural linguistic phenomenon in the same way that the spelling patterns activities were. If the children's period of participation were longer than what it was, I would have liked to set aside time for the children to critically evaluate their peer's programs and make modifications to the programs, as they had done with my English spelling pattern program.

In general, the three children's programs got a lot of positive, excited response from the other children, who were interested in playing with the programs and finding out how they were made. I feel very strongly that given more time, many of the other children would have figured out what they wanted to work on. In no way do I see the group of children who didn't come up with ideas for their own computer programs as incapable of doing so. A total of six to nine hours, spanning a period of three weeks, was a very short period of time to expect all children to become inspired with their own ideas. I also hope that as I gain more experience in working with children, I will become better at developing a discourse with children to help them identify their areas of interest.

A "loms" generator ★

RI (age 10, grade 4) wanted to make a program that would generate words in "an alien language." In the alien language, the syllable is not made up of an onset, a vowel, and a coda like in English. Instead, the syllable consists of a "farlor", a "chores", a "turtol", and a "kahoused". When I asked RI what all these terms meant, he explained that a "turtol" is like a vowel, "farlor" and "chores" are like onsets ("in this language, you need two onsets"), and a "kahoused" is like a coda. I asked RI to explain why this alien language needed two onsets. Instead of giving a verbal explanation, RI drew four boxes on a sheet of paper, labeled them "farlor", "chores", "turtol" and "kahoused", and wrote some letters in each of the four boxes. RI had written a handful of two- and three-letter consonant combinations inside the "farlor", "chores", and "kahoused" boxes, and he had written *a*, *e*, *i*, *o*, and *u* inside the "turtol" box. "Here I'll show you," RI said to me, "This is how a word is made in the alien language." He then wrote *flzhash* on the piece of paper and explained that the *fl* is a "farlor" ("it was picked from the box with all the 'farlors' in it), the *zh* is a

"chores", the *a* is a "turtol", and the *sh* is a "kahoused". From this it became apparent to me that RI's definition of an onset was a consonant combination with two or three letters that occupy a position before the vowel – which is exactly what an onset is in the English language.[39] The alien language needed to have two onsets because RI wanted words in the language to have four to six letters before the vowel. I worked with RI on writing a simple procedure that would generate a "loms" – which is the word for "word" in RI's alien language – by picking at random an element from "farlor", "chores", "turtol", and "kahoused". RI explained that a "loms" in the alien language means both a word and a syllable, because there is only one syllable in each "loms". This is the procedure we wrote for generating "loms":

```
to loms
insert textpick "FARLOR
insert textpick "CHORES
insert textpick "TURTOL
insert textpick "KAHOUSED
end
```

The design of RI's "loms" generator clearly demonstrated RI's developing understanding of the formal concept of syllable subunits. I asked RI to show his program (Figure 12) to his group. They were amused by "all the funny-looking words" generated by the program.

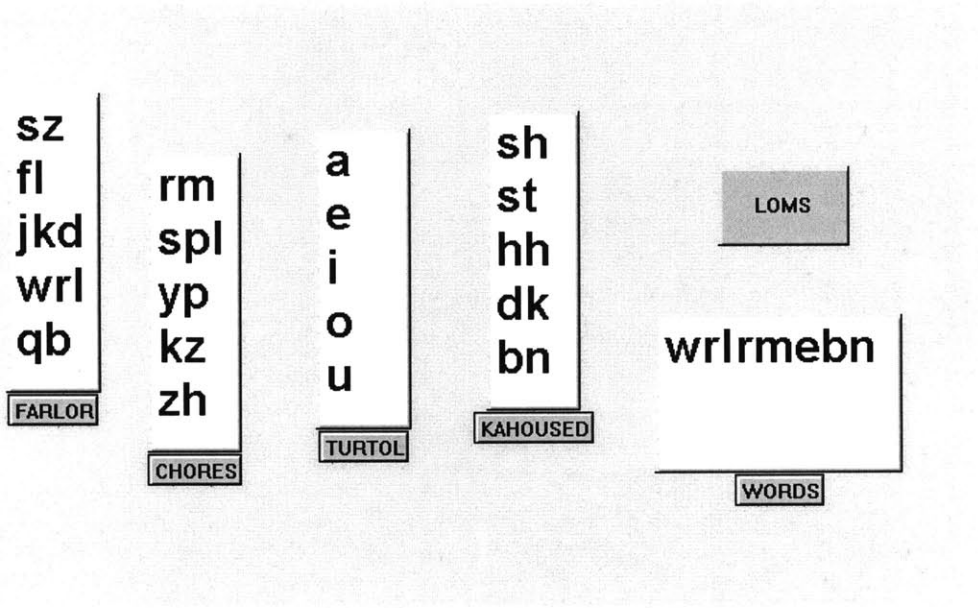


Figure 12 RI's "loms" generator

[39] Kenstowicz94.

Spelling with Winnie-the-Pooh ★

While GA, HI (both age 9, grade 3), and LA (age 9, grade 4) were thinking about what to do for their computer programs, they saw a copy of A. A. Milne's *The Complete Tales of Winnie-the-Pooh* in my office and came upon an idea. The group flipped through the book and found what they were looking for. "Look," they directed my attention to the book, "Pooh spells honey '*hunny*'." [40] The group then decided that they wanted to work on a computer program that would translate a correctly spelled English word into "how Pooh would spell it."

The group first worked on making a list of English words spelled correctly and of Pooh's spelling of those words. GA, HI, and LA took turns coming up with words at random, thinking about how they are spelled "the correct way", then thinking about how the words could be spelled incorrectly.

After they were done with their list, I helped them write a program for translating correctly spelled English words into Pooh's spelling of those words. The translating program (Figure 13) has two textboxes. The textbox named "ENGLISH" serves as the input textbox where correctly spelled English words are typed in, and the textbox named "POOH" serves as the output textbox where the program reports the words translated into Pooh's spelling when the "TRANSLATE" button is clicked on.[41]

[40] Milne94, pg. 59.

[41] The entire set of procedures for the "spelling with Winnie-the-Pooh" program can be found in the appendix.

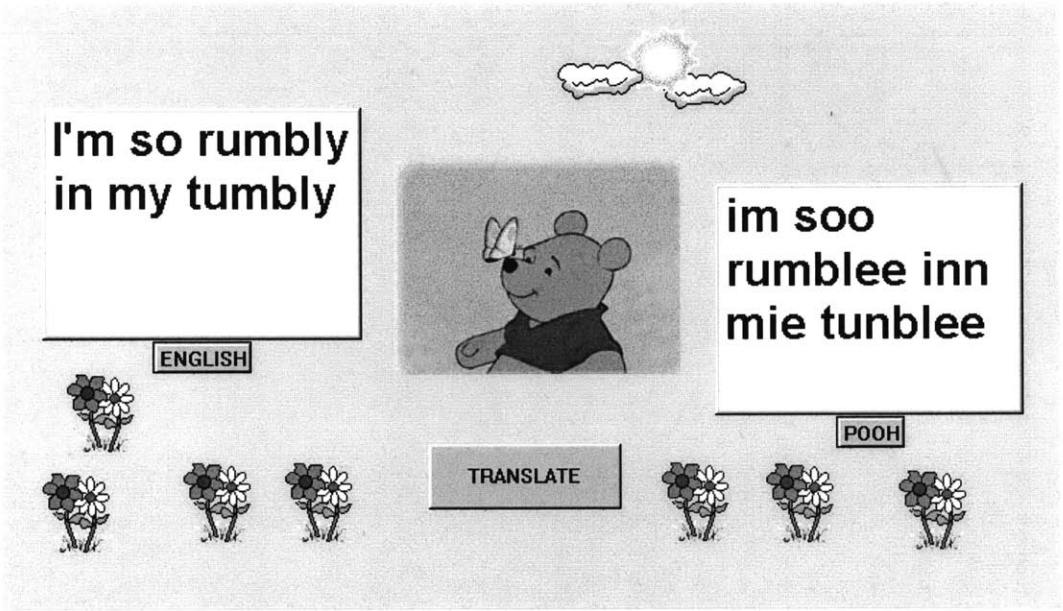


Figure 13 Spelling with Winnie-the-Pooh

A secret language ★

UT (age 10, grade 5) wanted to make a program that would transform an English sentence into a secret language. To transform English into her secret language, UT explained, you take off the first letter of each word and stick it at the end of the word, then you add the letter *a*. "It's basically like pig latin, but a little different," she explained. I helped UT write a procedure that works exactly like her explanation of the transformation of the English word.

```
to transform :word
SOLUTION,
insert butfirst :word insert first :word insert "a
end
```

To get the program to transform each word in an input sentence, I helped UT write a superprocedure called **do-it** which parses the input sentence into words and runs the **transform** procedure iteratively until each word in the input sentence is transformed.

```
to do-it
make "n 1
repeat count parse ENGLISH
       [transform item :n parse ENGLISH
        SOLUTION, insert "| |
        make "n :n + 1]
end
```

UT used her program (Figure 14) to write short sentences, which were transformed into her secret language. She would then challenge the other children in her group to figure out what the sentence said (she would make the "ENGLISH" textbox invisible before she put out her challenge).

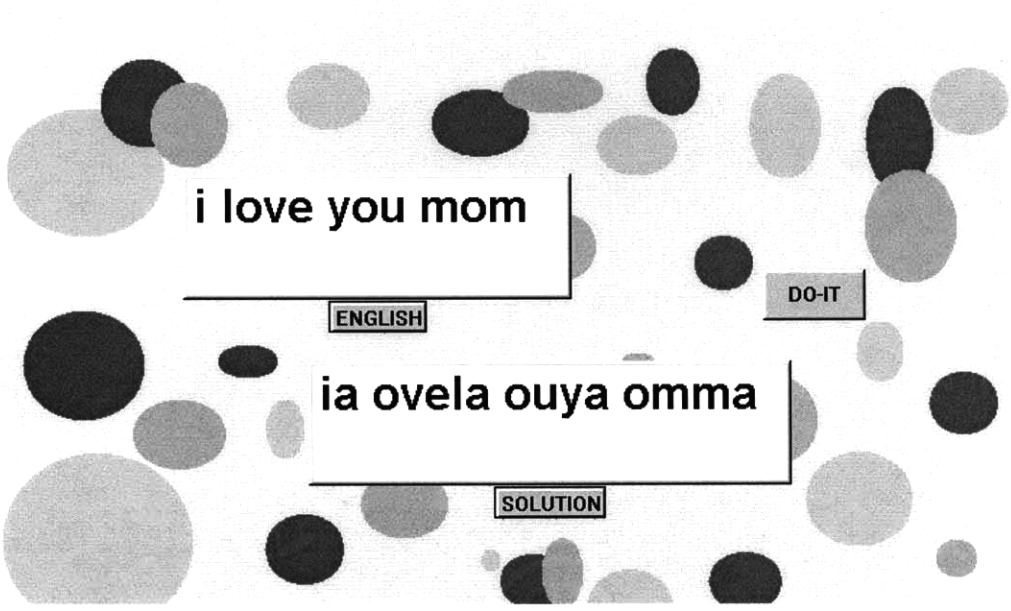


Figure 14 UT's secret language

KS (age 8, grade 3) liked UT's program so much; she wanted to make one of her own. Using her program as a template, UT was able to help KS make a program similar to hers – KS's transformation of the English word differs from UT's transformation in that *af* was added (instead of *a*) to the end of each word.

CONCLUSIONS

At the center of this thesis work lies my belief that children's intuitions about the grammar of their language are resources, which children can use to leverage understanding of formal grammatical concepts. In working to confirm this belief, I used computer programming to provide a meaningful context in which formal grammatical concepts were introduced to children through the engagement and mobilization of their linguistic intuitions.

My thesis work was initiated with a computer program that I made about the English spelling pattern. Embodied in my computer program is the formal notion that spelling patterns arise from phonological constraints governing the syllable. Children were able to use their intuitions about possible and impossible words in English to come up with an intuitive explanation for the regularities observed in the computer program. Children's intuitive understanding of my program was then used as a foundation on which I explained in formal terms the linguistic concepts embedded in my program.

It was very important that children's intuitive understanding, not the formalisms, came first. In this way, my explanations of formal concepts like "syllable", "onset", and "coda" were connected to the children's concrete experience with my program and to their intuitive understanding of my program. It was from this connection that the formal concepts derived their meaning and purpose. Children could see how I had used these formal concepts for descriptive purposes in my program and how they themselves could use the formal concepts to deepen the understanding which they had reached intuitively.

In their critical evaluation and modification of my program, children also used their intuitions as a starting point for gaining comprehension on how my program worked, how they wanted to make it work differently, and how they could implement their ideas for program modification. Children further advanced their understanding of formal concepts describing the phenomenon of spelling patterns when they used my program as a template for extending the generative model to Malay and Polish, and to their own invented languages.

In essence, formal grammatical concepts like "syllable", "onset", and "coda" were appropriated by the children because these concepts became meaningful to children in the context of their intuitions about language, and also because these concepts became purposeful to children in the context of programming the computer. In other words, children developed an understanding for the formal concepts because they could relate the concepts to what they already knew, and they could use the concepts to understand how something was made or to make something themselves. There is also a positive feedback loop at work here: in using the formal concepts to understand how something was made and to make something themselves, children further deepened their understanding of the formal grammatical concepts.

During a period of eight weeks, twenty-one children took part in and contributed to an intellectual culture in which formal grammatical concepts were not something valuable in and of themselves, but were valued to the extent that they could be meaningfully understood and purposefully used to deepen one's understanding of language and to articulate and make tangible one's ideas about language. In this intellectual culture, children discovered that they could make sense of English spelling after all, that they do understand aspects of grammar, and that they are quite good at thinking about language. More importantly, children gained confidence in their ability to be their own intellectual agents and developed an interest in further exercising their intelligence in the scientific domain of linguistic inquiry.

REFERENCES

[Chomsky72] Chomsky, Noam. *Language and Mind.* New York: Harcourt Brace Jovanovich, 1972.

[Chomsky68] Chomsky, Noam and Halle, Morris. *The Sound Pattern of English.* New York: Harper & Row, 1968.

[Ferreiro82] Ferreiro, Emilia and Teberosky, Ana. *Literacy Before Schooling.* Portsmouth: Heinemann Educational Books, Inc., 1982.

[Goldenberg87] Goldenberg, Ernest Paul and Feurzeig, Wallace. *Exploring Language with Logo.* Cambridge: MIT Press, 1987.

[Karmiloff-Smith79] Karmiloff-Smith, Annette. *A Functional Approach to Child Language.* Cambridge: Cambridge University Press, 1979.

[Karmiloff-Smith92] Karmiloff-Smith, Annette. *Beyond Modularity: A Developmental Perspective on Cognitive Science.* Cambridge: MIT Press, 1992.

[Kenstowicz94] Kenstowicz, Michael. *Phonology in Generative Grammar.* Cambridge: Blackwell Publishers, 1994.

[Milne94] Milne, A. A. *The Complete Tales of Winnie-the-Pooh.* New York: Dutton Children's Books, 1994.

[Papert80] Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, 1980.

[Pinker94] Pinker, Steven. *The Language Instinct.* New York: William Morrow & Co., 1994.

[Resnick92] Resnick, Mitchel. *Beyond the Centralized Mindset: Explorations in Massively-Parallel Microworlds.* MIT Media Laboratory Doctoral Dissertation, 1992.

[Schön81] Schön, Donald. *Intuitive Thinking? A Metaphor Underlying Some Ideas of Educational Reform.* Cambridge: Division for Study and Research in Education, 1981.

APPENDIX

Questionnaire ★

Children were asked to circle a number for each set of statements, based on how they feel about what each set of statements says. For example, if they strongly agree with the statement on the left "I like school", then they should circle 5. If they strongly agree with the statement on the right "I don't like school", then they should circle 1. If they feel somewhere in between the two statements, they can decide the degree to which they agree with either statement.

I like school. I don't like school.

5 4 3 2 1

I like English class. I don't like English class.

5 4 3 2 1

I like doing English homework. I don't like doing English homework.

5 4 3 2 1

I like reading books. I don't like reading books.

5 4 3 2 1

I like writing. I don't like writing.

5 4 3 2 1

I am good at spelling. I am not good at spelling.

5 4 3 2 1

I understand grammar. I don't understand grammar.

5 4 3 2 1

I like thinking about language. I don't like thinking about language.

5 4 3 2 1

I like computers. I don't like computers.

5 4 3 2 1

Procedures for my English spelling pattern program ★

```
to startup
setvowels
       [a e i o u
        ae ai ao au
        ea ee ei eo eu
        ia ie io iu
        oa oe oi oo ou
        ua ue ui uo]
setconsonants
       [b c d f g h j k l m n p q r s t v w x y z]
setonsets
       [b c d f g h j k l m n p qu r s t v w x y z
        bl br
        ch cl cr
        dr dw
        fl fr
        gh gl gr gw
        kl kr kw
        ph pl pr
        sc sh sk sl sm sn sp squ st sw
        th tr tw
        wh wr
        chr
        scl scr shl shr skl skr spl spr str
        thr thw]
setcodas
       [b d f g k l m n p r s t v w x y z
        dd ff gg ll rr ss tt zz
        ch ck
        ft
        ld lf lk lm lp lt
        mb mn mp
        nd ng nk nt
        ph pt
        rb rd rf rk rl rm rn rp rt rv
        sh sk sm sp st
        th
        wk wl
        bs ds gs ks ls ms ns ps rs ts ws ys]
end

to check
let [new-word get "input "text]
ifelse (count :new-word) < 5
       [parse-syl get "input "text
        evaluate-syl
        evaluate-syl2]
       [setc "red setresult [Too long!]]
end
```

```
to parse-syl :syllable
if not empty? :syllable
[make "letter first :syllable
ifelse equal? "q :letter
  [deal-with-q]
  [ifelse and empty? vowel
              not member? :letter parse vowels
    [onset, insert :letter
     parse-syl butfirst :syllable]
    [ifelse and not empty? vowel
                 not member? :letter parse vowels
      [coda, insert :letter
       parse-syl butfirst :syllable]
       [ifelse and (member? :letter parse vowels)
                   (not empty? coda)
         [setsyl2 last coda
         setcoda butlast coda
         syl2, insert :syllable
          parse-syl2 syl2]
         [vowel, insert :letter
         parse-syl butfirst :syllable]]]]]
end

to evaluate-syl
ifelse not (and
            (not empty? onset)
            (not empty? coda)
            (not empty? onset2))
  [ifelse or (member? onset parse allowed-onsets)
             (empty? onset)
    [ifelse (member? vowel parse vowels)
      [ifelse or (member? coda parse allowed-codas)
                 (empty? coda)
        [setc "green setresult [I like it!]]
        [setc "red setresult [I don't like it.]]]
       [setc "red setresult [I don't like it.]]]
    [setc "red setresult [I don't like it.]]]
  [setc "red setresult [I don't like it.]]
end

to deal-with-q
onset, insert first :syllable
onset, insert first butfirst :syllable
ifelse member? first (butfirst butfirst :syllable) parse vowels
       [parse-syl butfirst butfirst :syllable]
       [syl-recur butfirst butfirst :syllable
        vowel, cleartext]
end
```

```
to parse-syl2 :syllable
if not empty? :syllable
[make "letter first :syllable
ifelse equal? "q :letter
  [deal-with-q2]
  [ifelse and empty? vowel2
              not member? :letter parse vowels
    [onset2, insert :letter
     parse-syl2 butfirst :syllable]
    [ifelse and not empty? vowel2
                not member? :letter parse vowels
      [coda2, insert :letter
       parse-syl2 butfirst :syllable]
      [if member? :initial vowels
        [vowel2, insert :letter
         parse-syl2 butfirst :syllable]]]]
end

to evaluate-syl2
if not (and
        (empty? onset2)
        (empty? coda2)
        (empty? vowel2))
  [ifelse and (member? onset2 parse allowed-onsets)
              (not empty? onset2)
    [ifelse and (member? vowel2 parse vowels)
                (not empty? vowel2)
      [ifelse or (member? coda2 parse allowed-codas)
                 (empty? coda2)
        [setc "green setresult [I like it!]]
        [setc "red setresult [I don't like it.]]]
      [setc "red setresult [I don't like it.]]]
    [setc "red setresult [I don't like it.]]]
end

to deal-with-q2
onset2, insert first :syllable
onset2, insert first butfirst :syllable
ifelse member? first (butfirst butfirst :syllable) parse vowels
       [parse-syl2 butfirst butfirst :syllable]
       [syl-recur2 butfirst butfirst :syllable
        vowel, cleartext]
end

to new-word
input, cleartext
onset, cleartext
onset2, cleartext
vowel, cleartext
vowel2, cleartext
coda, cleartext
coda2, cleartext
result, cleartext
end
```

Children's lists of Malay and Polish words ★

Malay words:

Ahad	anging	baik	bola	dua	enam	hidung	isnin
Jauh	jumaat	kakak	kalian	kasih	kawan	kelua	khamis
Kucing	maaf	mac	mak	mei	pokok	rambut	sabtu
Selamat	sifar	terima	tidak	tinggal	ya		

Polish words:

Basen	czas	cztery	dla
Dobranoc	dobrze	dziesiec	dziewiec
Dworzec	grudzien	jeden	jutro
Kwiecien	matka	most	ojciec
Paszport	piec	pociag	poczta
Pokoj	polski	prosze	przyjazd
Siedem	sklep	sroda	stacja
Szesc	szkol	szpital	toalety
Ulica	wrzesien	wtorek	wzgorze
Zima			

Procedures for the "spelling with Winnie-the-Pooh" program ★

```
to check :word
if equal? :word "honey [insert [hunny]]
if equal? :word "snake [insert [snac]]
if equal? :word "eye [insert [i]]
if equal? :word "you [insert [u]]
if equal? :word "owl [insert [wol]]
if equal? :word "tummy [insert [tunny]]
if equal? :word "please [insert [plez]]
if equal? :word "knock [insert [nok]]
if equal? :word "I [insert [i]]
if equal? :word "hello [insert [helow]]
if equal? :word "of [insert [ov]]
if equal? :word "big [insert [bij]]
if equal? :word "can [insert [kan]]
if equal? :word "read [insert [reed]]
if equal? :word "duck [insert [duk]]
if equal? :word "sun [insert [som]]
if equal? :word "roo [insert [ru]]
if equal? :word "unexpected [insert [un ecpd]]
if equal? :word "eeyore [insert [e your]]
if equal? :word "tigger [insert [tigre]]
if equal? :word "dump [insert [demp]]
if equal? :word "glasses [insert [glacis]]
if equal? :word "heaven [insert [heben]]
if equal? :word "know [insert [cno]]
if equal? :word "tumbly[insert [tunblee]]
if equal? :word "my [insert [mie]]
if equal? :word "in [insert [inn]]
if equal? :word "rumbly [insert [rumblee]]
if equal? :word "so [insert [soo]]
if equal? :word "I'm [insert [im]]
insert "| |
end

to check-sentence
pooh, ct
make "n 1
repeat count parse english
       [check item :n parse english
        make "n :n + 1]
end
```